Husna Manalai
Assignment 4 report

MNIST Digit OCR

Test Accuracy Results:
FCNET: Average loss: 0.1177, Accuracy: 9629/10000 (96%) Model: FCNet
CONVNET: Average loss: 0.0282, Accuracy: 9904/10000 (99%) Model: ConvNet

The ConvNet achieved a higher accuracy (99%) compared to the FCNet (96%). This indicates that the ConvNet is more effective in recognizing handwritten digits in the MNIST dataset.
The ConvNet also achieved a lower average loss (0.0282) compared to the FCNet (0.1177). This further reinforces the superior performance of the ConvNet in terms of minimizing the prediction error.

Comparing the models:
The The fully connected network consists of fully connected layers:
Input layer: 784 neurons (flattened 28x28 images)
Hidden layer 1: 256 neurons
Hidden layer 2: 128 neurons
Output layer: 10 neurons (for 10 classes)
Activation Function: ReLU for hidden layers, Softmax for the output layer

The Convolutional network consists of two convolutional layers followed by fully connected layers:
Convolutional layer 1: 32 filters, 5x5 kernel
Convolutional layer 2: 64 filters, 5x5 kernel
Fully connected layer 1: 1024 neurons (flattened feature maps)
Fully connected layer 2: 256 neurons
Output layer: 10 neurons (for 10 classes)
Activation Function: ReLU for hidden layers, Softmax for the output layer
Pooling: Max pooling after each convolutional layer
The fully connected network, while effective, lacks the ability to capture spatial hierarchies and local patterns in the image data. It treats each pixel independently, which can limit its performance in tasks involving images.
Convolutional networks are specifically designed to handle image data. The convolutional layers are capable of detecting local patterns such as edges, textures, and shapes, and the pooling layers help in reducing the spatial dimensions, making the network more robust to variations in the input images.

Some of the reason why I think ConvNet Performed Better:
Spatial Hierarchies: ConvNets can capture spatial hierarchies in the data, which are crucial for understanding the structure of images. This allows ConvNets to extract more relevant features from the input images.

Parameter Sharing: The use of convolutional layers means that parameters (weights) are shared across different parts of the image, which helps in learning more generalized features and reduces the number of parameters.
Local Receptive Fields: Each neuron in a convolutional layer is connected only to a local region of the input, which helps in preserving spatial relationships and detecting patterns regardless of their position in the image.
Pooling Layers: Pooling layers reduce the spatial dimensions of the feature maps, which not only reduces the computational complexity but also makes the model more invariant to small translations and distortions in the input images.

CIFAR-10 (FCNET && CONVNET)

I also wanted to train a fully connected neural network and a convolutional neural network on the CIFAR-10 dataset.
And the results I got was that ConvNet achieved a higher accuracy (69.35%) compared to the FCNet (53.52%) again. This indicates that the ConvNet is more effective in recognizing objects in the CIFAR-10 and MNIST datasets. The ConvNet's ability to capture and utilize spatial information in the images led to higher accuracy and lower loss. The architectural advantages of ConvNets, such as spatial hierarchies, parameter sharing, local receptive fields, and pooling, make them more suitable for image classification tasks, especially with more complex datasets like CIFAR-10.

Custom Dataset
Bike Sharing Dataset(FCNet, and RNN)

Bike Sharing Dataset: At first I wanted to use the wine dataset however I realized the data was too small (about 1600 points) and so I found the Bike Sharing dataset. The Bike Sharing dataset is a publicly available dataset from the UCI Machine Learning Repository. It contains data on bike rental counts in a city over a span of two years. The dataset is useful for understanding factors affecting bike rentals, predicting future demand, and planning for bike distribution. The dataset includes 16 attributes and about 17389 points . The dataset is used to analyze and predict bike rental demand.

Results:
Fully Connected Network (FCNet) Test set: Average loss: 39.2963
Recurrent Neural Network (RNN) Test set: Average loss: 554.7947

Fully Connected Network (FCNet) Architecture:
Input Layer: Takes the input features (118 dimensions after preprocessing).
Hidden Layer 1: Fully connected layer with 256 neurons.
Hidden Layer 2: Fully connected layer with 128 neurons.
Hidden Layer 3: Fully connected layer with 64 neurons.
Output Layer: Fully connected layer with 1 neuron (for regression).

Recurrent Neural Network (RNN) Architecture:
Input Layer: Takes the input features (118 dimensions after preprocessing).
LSTM Layer: LSTM layer with 64 hidden units and 2 layers.
Fully Connected Layer: Linear layer with 1 output for regression.

The FCNet achieved a significantly lower average loss (39.2963) compared to the RNN (554.7947), indicating better performance in predicting bike rental counts.

Model Performance:
FCNet: Despite its simplicity, the FCNet performed well. The network's fully connected layers effectively captured the relationships between input features and the target variable without needing sequential modeling.
RNN: The RNN struggled with this regression task. While RNNs are designed to handle sequential data, the bike sharing dataset may not exhibit strong temporal dependencies that the RNN could leverage, leading to poorer performance.

Why FCNet Performed Better:
Nature of Data: The bike sharing dataset, although containing time-related features, does not strongly require sequential processing. The FCNet's fully connected layers were sufficient to model the data.
Model Complexity: The FCNet's architecture is straightforward, mapping input features directly to the output. This simplicity may have contributed to its better performance, avoiding the complexity and potential overfitting associated with RNNs.
Sequential Dependence: The RNN, particularly the LSTM used, is designed to capture temporal dependencies and sequence-related patterns. However, the dataset may not have strong sequential patterns for the RNN to exploit, leading to higher average loss.

So the architectural simplicity of the FCNet allowed it to capture the necessary patterns more effectively. On the other hand, the RNN, designed for sequential data, did not find the same level of relevant temporal dependencies in this dataset, leading to poorer performance.

Additional Models:

I also test the Bike Sharing dataset with the:
GRU with Convolutional Layers and Dropout (2 Hidden GRU Layers) and a Transformer Encoder with Batch Normalization.

The Results:
GRU with Convolutional Layers Test set: Average loss: 607.0520
Transformer Encoder with Batch Normalization and ReLU Test set: Average loss: 55.6702

The Transformer Encoder with Batch Normalization and ReLU achieved a significantly lower average loss (55.6702) compared to the GRU with Convolutional Layers (607.0520). This indicates that the Transformer model is more effective in predicting bike rental counts.

Model Performance:
GRU with Convolutional Layers: This model struggled with the regression task on the bike sharing dataset, as indicated by the higher average loss. While GRU layers are designed to handle sequential data and convolutional layers can capture local patterns, the combination did not yield the desired accuracy for this dataset.
Transformer Encoder: The Transformer model, which excels in capturing long-range dependencies and interactions between features, performed significantly better. The use of batch normalization helped stabilize training, and the ReLU activation provided non-linearity, allowing the model to capture complex patterns in the data.

Why I Think Transformer Performed Better:
Nature of Data: The bike sharing dataset includes features related to time, weather, and other factors that may have complex interactions. The Transformer's self-attention mechanism is well-suited for capturing these interactions, leading to better performance.
Batch Normalization: Batch normalization helps in stabilizing and accelerating training, which can lead to better generalization and lower test loss.
Architectural Advantages: The Transformer's ability to handle dependencies between all pairs of input features allows it to model relationships more effectively than GRU layers, which are inherently sequential.

Feature Engineering and Input Dimensions:
Preprocessing: Both models were provided with extensively engineered features, including polynomial features, interaction terms, and statistical features, standardized to zero mean and unit variance.
Padding for Input Dimensions: Ensuring the input dimension is divisible by the number of heads in the Transformer model was essential for its operation and contributed to the improved performance.

In summary, the results highlight the importance of matching the model architecture to the dataset's characteristics. The Transformer model's architectural advantages, including self-attention mechanisms and batch normalization, made it more suitable for predicting bike rental counts in this dataset.